Electronics: principles of digital electronics

Module by: Free High School Science Texts Project.

The Principles of Digital Electronics

The circuits and components we have discussed are very useful. You can build a radio or television with them. You can make a telephone. Even if that was all there was to electronics, it would still be very useful. However, the great breakthrough in the last fifty years or so has been in **digital** electronics. This is the subject which gave us the computer. The computer has revolutionized the way business, engineering and science are done. Small computers programmed to do a specific job (called microprocessors) are now used in almost every electronic machine from cars to washing machines. Computers have also changed the way we communicate. We used to have telegraph or telephone wires passing up and down a country — each one carrying one telephone call or signal. We now have optic fibres each capable of carrying tens of thousands of telephone calls using **digital** signals.

So, what is a digital signal? Look at **Figure 1** (#uid85). A normal signal, called an **analogue** signal, carries a smooth wave. At any time, the voltage of the signal could take any value. It could be 2,00 V or 3,53 V or anything else. A digital signal can only take certain voltages. The simplest case is shown in the figure — the voltage takes one of two values. It is either **high**, or it is **low**. It never has any other value.

These two special voltages are given symbols. The low voltage level is written 0, while the high voltage level is written as 1. When you send a digital signal, you set the voltage you want (0 or 1), then keep this fixed for a fixed amount of time (for example 0.01 μs), then you send the next 0 or 1. The digital signal in **Figure 1** (#uid85) could be written 01100101.

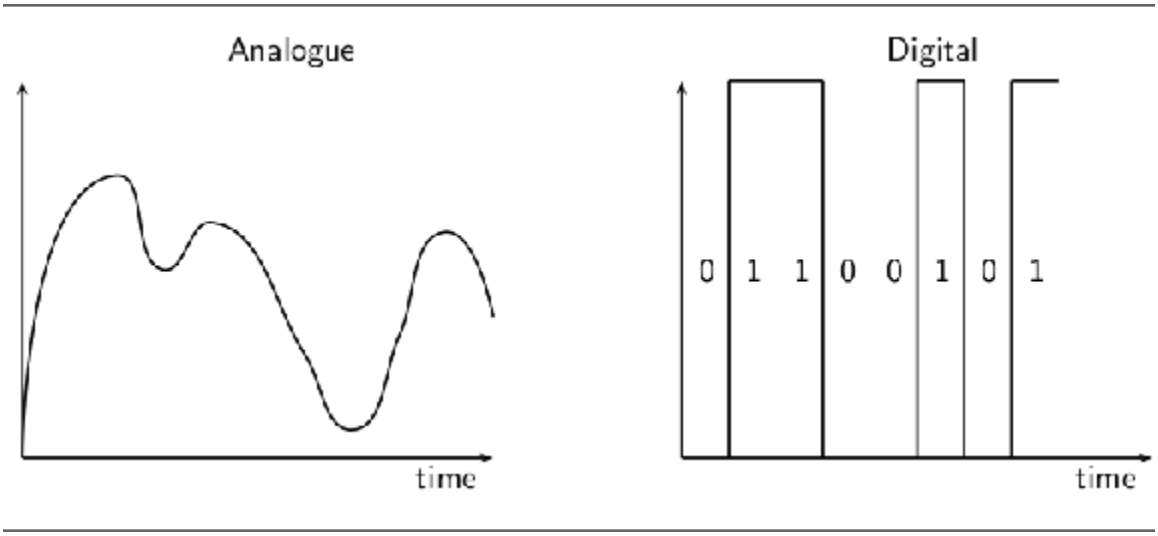

Figure 1: The difference between normal (analogue) signals and digital signals.

Why are digital signals so good?

1. Using a computer, any information can be turned into a pattern of 0s and 1s. Pictures, recorded music, text and motion pictures can all be turned into a string of 0s and 1s and transmitted or stored in the same way. The computer receiving the signal at the other end converts it back again. A Compact Disc (CD) for example, can store music or text or pictures, and all can be read using a computer.
2. The 0 and the 1 look very different. You can immediately tell if a 0 or a 1 is being sent. Even if there is interference, you can still tell whether the sender sent a 0 or a 1. This means that fewer mistakes are made when reading a digital signal. This is why the best music recording technologies, and the most modern cameras, for example, all use digital technology.
3. Using the 0s and 1s you can count, and do all kinds of mathematics. This will be explained in more detail in the next section.

The simplest digital circuits are called **logic gates**. Each logic gate makes a decision based on the information it receives. Different logic gates are set up to make the decisions in different ways. Each logic gate will be made of many microscopic transistors connected together within a thin wafer of silicon. This tiny circuit is called an Integrated Circuit or I.C. - all the parts are in one place (integrated) on the silicon wafer.

Logic Gates

There are five main types of logic gate: NOT, AND, OR, NAND and NOR. Each one makes its decision in a different way.

The NOT Gate

Problem: You want an automatic circuit in your office to turn on the heating in the winter. You already have a digital electronic temperature sensor. When the temperature is high, it sends out a 1. When the office is cold, it sends out a 0. If this signal were sent straight to the heater, the heater would turn on (1) when it was already hot, and would stay off when it was cold. This is wrong! To make the heater work, we need a circuit which will change a 0 (from the sensor) into a 1 (to send to the heater). This will make the heater come on when it is cold. You also want it to change a 1 (from the sensor) into a 0 (to send to the heater). This will turn the heater off when the room is hot. This circuit is called an **inverter** or **NOT gate**. It changes 0 into 1 (1 is NOT 0). It changes 1 into 0 (0 is

NOT 1). It changes a signal into what it is NOT.

The symbol for the NOT gate is:

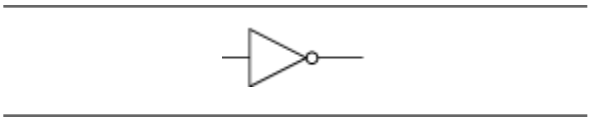

Figure 2

The action of the NOT gate can be written in a table called a **truth table**. The left column shows the possible inputs on different rows. The right column shows what the output (decision) of the circuit will be for that input. The truth table for the NOT gate is shown below.

Input	Output
0	1
1	0

TABLE 1

When you read the truth table, the top row says, "If the input is 0, the output will be 1." For our heater, this means, "If the room is cold, the heater will turn on." The bottom row says, "If the input is 1, the output will be 0." For our heater, this means, "If the room is hot, the heater will switch off."

The AND Gate

Problem: An airliner has two toilets. Passengers get annoyed if they get up from their seat only to find that both toilets are being used and they have to go back to their seat and wait. You want to fit an automatic circuit to light up a display if both toilets are in use. Then passengers know that if the light is off, there will be a free toilet for them to use. There is a sensor in each toilet. It gives out a 0 of the toilet is free, and a 1 if it is in use. You want to send a 1 to the display unit if **both** sensors are sending 1s. To do this, you use an AND gate.

The symbol for the AND gate is:

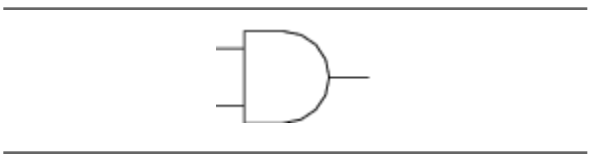

Figure 3: Symbol for the AND logic gate.

The truth table for the AND gate is shown below. An AND gate has two inputs (the NOT gate only had one). This means we need four rows in the truth table, one for each possible set of inputs. The first row, for example, tells us what the AND gate will do if both inputs are 0. In our airliner, this means that both toilets are free. The right column has a 0 showing that the output will be 0, so the display will not light up. The second row has inputs of 0 and 1 (the first toilet is free, the other is in use). Again the output is 0. The third row tells us what will happen if the inputs are 1 and 0 (the first toilet is in use, and the second is free). Finally, the last line tells us what will happen if both inputs are 1 (the first toilet is in use and the second toilet is in use). It is only in this case that the output is 1 and the display lights up.

Inputs		Output
A	B	
0	0	0
0	1	0
1	0	0
1	1	1

TABLE 2

This device is called an AND gate, because the output is only 1 if one input AND the other input are both 1.

Using 0 and 1 to mean True and False

When we use logic gates we use the low voltage state 0 to represent `false'. The high voltage state 1 represents `true'. This is why the word AND is so appropriate. A AND B is true (1) if, and only if, A is true (1) AND B is true (1).

AND and multiplication

Sometimes, the AND operation is written as multiplication. A AND B is written AB. If either A or B are 0, then AB will also be 0. For AB to be 1, we need A and B to both be 1. Multiplication of the numbers 0 and 1 does exactly the same job as an AND gate.

The NAND Gate

Problem: You build the circuit for the airliner toilets using an AND gate. Your customer is pleased, but she says that it would be better if the display lit up when there **was** a free toilet. In other words, the display should light up unless both toilets are in use. To do this we want a circuit which does the opposite of an AND gate. We want a circuit which would give the output 0 where an AND gate would give 1. We want a circuit which would give the output 1 where an AND gate would give 0. This circuit is called a NAND gate.

The symbol for the NAND gate is:

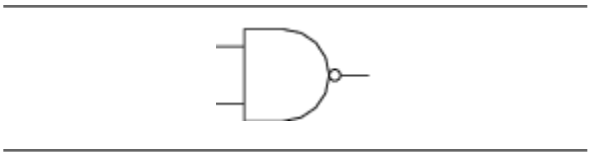

Figure 4

The truth table for the NAND gate is shown below.

Inputs		Output
A	B	
0	0	1

TABLE 3

0	1	1
1	0	1
1	1	0

You may have noticed that we could have done this job on the airliner by using our earlier circuit, with a NOT gate added between the original AND gate and the display. This is where the word NAND comes from — it is short for NotAND.

The OR Gate

Problem: A long, dark corridor has two light switches — one at each end of the corridor. The switches each send an output of 0 to the control unit if no-one has pressed the switch. If someone presses the switch, its output is 1. The lights in the corridor should come on if either switch is pressed. To do this job, the control unit needs an OR gate. The symbol for the OR gate is:

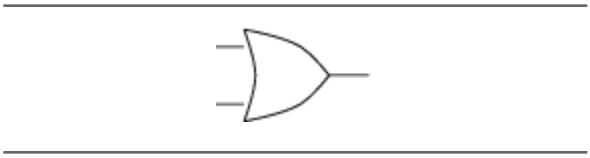

Figure 5: Symbol for the OR logic gate.

The truth table for the OR gate is shown.

Inputs		Output
A	B	
0	0	0
0	1	1
1	0	1
1	1	1

TABLE 4

You can see that the output is 1 (and the lights come on in the corridor) if either one switch OR the other is pressed. Pressing both switches also turns on the lights, as the last row in the table shows.

OR and addition

Sometimes you will see A OR B written mathematically as A+B. This makes sense, since if A=0 and B=0, then A OR B = A+B = 0. Similarly, if A=0 and B=1, then A OR B = A+B = 1. If A=1 and B=0, then A OR B = A+B = 1 once again. The only case where the OR function differs from normal addition is when A=1 and B=1. Here A OR B = 1 in logic, but A+B=2 in arithmetic. However, there is no such thing as `2' in logic, so we define + to mean `OR', and write 1+1=1 with impunity!

If you wish, you can prove that the normal rules of algebra still work using this notation: A+(B+C) = (A+B)+C, A(BC) = (AB)C, and A(B+C) = AB + AC. This special kind of algebra where variables can only be 0 (representing false) or 1 (representing true) is called Boolean algebra.

The NOR Gate

The last gate you need to know is the NOR gate. This is opposite to the OR gate. The output is 1 if both inputs are 0. In other words, the output switches on if neither the first NOR the second input is 1. The symbol for the NOR gate is:

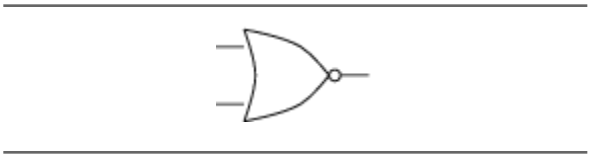

Figure 6: Symbol for the NOR logic gate.

The truth table for the NOR gate is shown below.

Inputs		Output
A	B	
0	0	1
0	1	0
1	0	0
1	1	0

TABLE 5

The examples given were easy. Each job only needed one logic gate. However any `decision making' circuit can be built with logic gates, no matter how complicated the decision. Here is an example.

EXERCISE 1: An Economic Heating Control

A sensor in a building detects whether a room is being used. If it is empty, the output is 0, if it is in use, the output is 1. Another sensor measures the temperature of the room. If it is cold, the output is 0. If it is hot, the output is 1. The heating comes on if it receives a 1. Design a control circuit so that the heating only comes on if the room is in use and it is cold.

SOLUTION

Step 1. **Think about what each sensor does :**

The first sensor tells us whether the room is occupied. The second sensor tells us whether the room is hot. The heating must come on if the room is occupied AND cold. This means that the heating should come on if the room is occupied AND (NOT hot).

Step 2. **Decide which gates to use and draw the circuit :**

To build the circuit, we first attach a NOT gate to the output of the temperature sensor. This output of the NOT gate will be 1 only if the room is cold. We then attach this output to an AND gate, together with the output from the other sensor. The output of the AND gate will only be 1 if the room is occupied AND the output of the NOT gate is also 1. So the heating will only come on if the room is in use and is cold. The circuit is shown below.

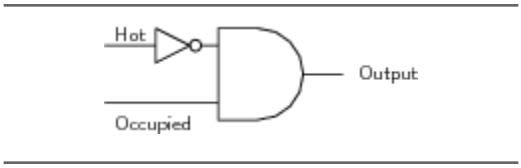

Figure 7

EXERCISE 2: Solving a circuit with two logic gates

Compile the truth table for the circuit below.

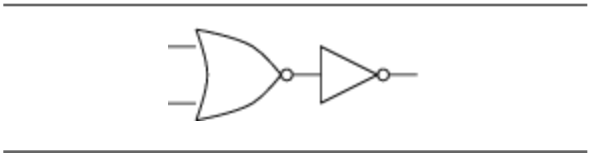

Figure 8

SOLUTION

Step 1. **Firstly, we label the inputs A and B. We also label the point where the two gates are connected C. :**

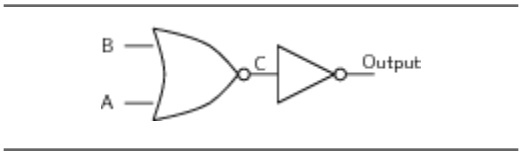

Figure 9

Step 2. **Next we prepare a truth table. :**

There is a column for each of the inputs, for the intermediate point C and also for the output. The truth table has four rows, since there are four possible inputs — 00, 01, 10 and 11.

A	B	C	Output
0	0		
0	1		
1	0		
1	1		

TABLE 6

Next we fill in the C column given that we know what a NOR gate does.

A	B	C	Output
0	0	1	
0	1	0	

TABLE 7

1	0	0	
1	1	0	

Next, we can fill in the output, since it will always be the opposite of C (because of the NOT gate).

A	B	C	Output
0	0	1	0
0	1	0	1
1	0	0	1
1	1	0	1

TABLE 8

Step 3. **Write the final answer :**

Finally we see that this combination of gates does the same job as an OR gate.

Each logic gate is manufactured from two or more transistors. Other circuits can be made using logic gates, as we shall see in the next section. We shall show you how to count and store numbers using logic gates. This means that if you have enough transistors, and you connect them correctly to make the right logic gates, you can make circuits which count and store numbers.

In practice, the cheapest gate to manufacture is usually the NAND gate. Additionally, Charles Peirce showed that NAND gates alone (as well as NOR gates alone) can be used to reproduce all the other logic gates.

The Principles of Digital Electronics

1. Why is digital electronics important to modern technology and information processing?
2. What two symbols are used in digital electronics, to represent a “high" and a “low"? What is this system known as?
3. What is a logic gate?
4. What are the five main types of logic gates? Draw the symbol for each logic gate.
5. Write out the truth tables for each of the five logic gates.
6. Write out the truth table for the following circuit. Which single gate is this circuit equivalent to?

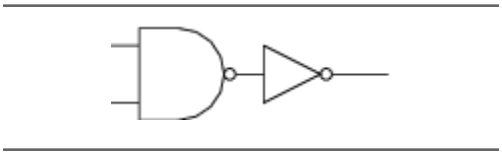

Figure 10

7. Write out the truth table for the following circuit. Which single gate is this circuit

equivalent to?

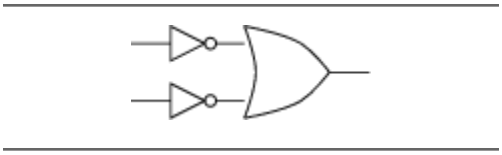

Figure 11

Using and Storing Binary Numbers

In the previous section, we saw how the numbers 0 and 1 could represent `false' and `true' and could be used in decision making. Often we want to program a computer to count with numbers. To do this we need a way of writing any number using nothing other than 0 and 1. When written in this way, numbers are called **binary numbers**.

DEFINITION 1: Binary Number System

A way of writing any number using only the digits 0 and 1.

Binary numbers

In normal (denary) numbers, we write 9+1 as 10. The fact that the `1' in 10 is the second digit from the right tells us that it actually means 10 and not 1. Similarly, the `3' in 365 represents 300 because it is the third digit from the right. You could write 365 as $3 \times 100 + 6 \times 10 + 5$. You will notice the pattern that the nth digit from the right represents 10^{n-1} . In binary, we use the nth digit from the right to represent 2^{n-1} . Thus 2 is written as 10 in binary. Similarly $2^2 = 4$ is written as 100 in binary, and $2^3 = 8$ is written as 1000 in binary.

EXERCISE 3: Conversion of Binary Numbers to Denary Numbers

Convert the binary number 10101 to its denary equivalent.

SOLUTION

Step 1. **Use what you know about converting numbers :**

We start on the right. The `1' on the right does indeed represent one. The next `1' is in the third place from the right, and represents $2^2 = 4$. The next `1' is in the fifth place from the right and represents $2^4 = 16$. Accordingly, the binary number 10101 represents 16+4+1 = 21 in denary notation.

EXERCISE 4: Conversion of Denary Numbers to Binary Numbers

Convert the decimal number 12 to its binary equivalent.

SOLUTION

Step 1. **Rewrite as a sum of powers and convert :**

Firstly we write 12 as a sum of powers of 2, so 12 = 8+4. In binary, eight is 1000, and four is 100. This means that twelve = eight + four must be 1000+100 = 1100 in binary. You could also write 12 as $1 \times 8 + 1 \times 4 + 0 \times 2 + 0 \times 1 = 1100$ in binary.

NOTE: Interesting Fact :

How do you write numbers as a sum of powers of two? The first power of two (the largest) is the largest power of two which is not larger than the number you are working with. In our last example, where we wanted to know what twelve was in binary, the largest power of two which is not larger than 12 is 8. Thus 12 = 8 + something. By arithmetic, the `something' must be 4, and the largest power of two not larger than this is 4 exactly. Thus 12 = 8 + 4, and we have finished.

A more complicated example would be to write one hundred in binary. The largest power of two not larger than 100 is 64 (1000000 in binary). Subtracting 64 from 100 leaves 36. The largest power of two not larger than 36 is 32 (100000 in binary). Removing this leaves a remainder of 4, which is a power of two itself (100 in binary). Thus one hundred is 64 + 32 + 4, or in binary 1000000 + 100000 + 100 = 1100100.

Once a number is written in binary, it can be represented using the low and high voltage levels of digital electronics. We demonstrate how this is done by showing you how an electronic counter works.

Counting circuits

To make a counter you need several `T flip flops', sometimes called `divide by two' circuits. A T flip flop is a digital circuit which swaps its output (from 0 to 1 or from 1 to 0) whenever the input changes from 1 to 0. When the input changes from 0 to 1 it doesn't change its output. It is called a **flip flop** because it changes (flips or flops) each time it receives a pulse.

If you put a series of pulses 10101010 into a T flip flop, the result is 01100110. **Figure 12** (#uid108) makes this clearer.

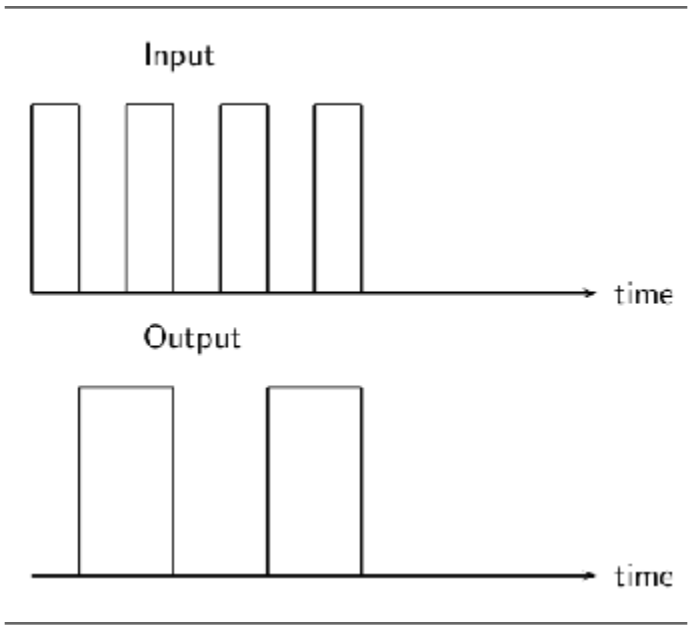

Figure 12: The output of a T flip flop, or `divide by two' circuit when a square wave is connected to the input. The output changes state when the input goes from 1 to 0.

As you can see from **Figure 12** (#uid108), there are half as many pulses in the output. This is why it is called a `divide by two' circuit.

If we connect T flip flops in a chain, then we make a counter which can count pulses. As an example, we connect three T flip flops in a chain. This is shown in **Figure 13** (#uid109).

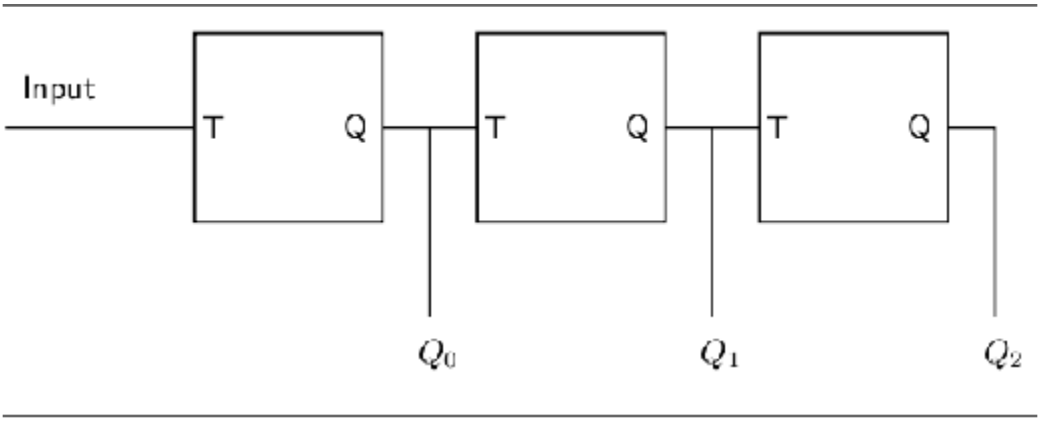

Figure 13: Three T flip flops connected together in a chain to make a counter. The input of each flip flop is labelled T, while each output is labelled Q. The pulses are connected to the input on the left. The outputs Q_0, Q_1 and Q_2 give the three digits of the binary number as the pulses are counted. This is explained in the text and in the next table.

When this circuit is fed with a stream of pulses, the outputs of the different stages change. The table below shows how this happens. Each row shows a different stage, with the first stage at the top. We assume that all of the flip flops have 0 as their output to start with.

Input	Output 1	Output 2	Output 3	Number of pulse	Number in binary
1	0	0	0	0	000

TABLE 9

0	1	0	0	1	001
1	1	0	0	1	001
0	0	1	0	2	010
1	0	1	0	2	010
0	1	1	0	3	011
1	1	1	0	3	011
0	0	0	1	4	100
1	0	0	1	4	100
0	1	0	1	5	101
1	1	0	1	5	101
0	0	1	1	6	110
1	0	1	1	6	110
0	1	1	1	7	111
1	1	1	1	7	111
0	0	0	0	8	1000
1	0	0	0	8	1000
0	1	0	0	9	1101
1	1	0	0	9	1101

The binary numbers in the right hand column count the pulses arriving at the input. You will notice that the output of the first flip flop gives the right most digit of the pulse count (in binary). The output of the second flip flop gives the second digit from the right (the `twos' digit) of the pulse count. The output of the third flip flop gives the third digit from the right (the `fours' digit) of the pulse count. As there are only three flip flops, there is nothing to provide the next digit (the `eights' digit), and so the eighth pulse is recorded as 000, not 1000.

This device is called a **modulo 8** counter because it can count in eight stages from 000 to 111 before it goes back to 000. If you put four flip flops in the counter, it will count in sixteen stages from 0000 to 1111, and it is called a modulo 16 counter because it counts in sixteen stages before going back to 0000.

DEFINITION 2: **Modulo**

The modulo of a counter tells you how many stages (or pulses) it receives before going back to 0 as its output. Thus a modulo 8 counter counts in eight stages 000, 001, 010, 011, 100, 101, 110, 111, then returns to 000 again.

NOTE: **Interesting Fact :**

If a counter contains n flip flops, it will be a modulo 2^n counter. It will count from 0 to $2^n - 1$.

Storing binary numbers

Counting is important. However, it is equally important to be able to remember the numbers. Computers can convert almost anything to a string of 0s and 1s, and therefore to a binary number. Unless this number can be stored in the computer's memory, the computer would be useless.

The memory in the computer contains many parts. Each part is able to store a single 0 or 1. Since 0 and 1 are the two binary digits, we say that each part of the memory stores one **bit**.

DEFINITION 3: Bit

One bit is a short way of saying one `binary digit'. It is a single 0 or 1.

NOTE: Interesting Fact :

If you have eight bits, you can store a binary number from 00000000 to 11111111 (0 to 255 in denary). This gives you enough permutations of 0s and 1s to have one for each letter of the alphabet (in upper and lower case), each digit from 0 to 9, each punctuation mark and each control code used by a computer in storing a document. When you type text into a word processor, each character is stored as a set of eight bits. Each set of eight bits is called a **byte**. Computer memories are graded according to how many bytes they store. There are 1024 bytes in a kilobyte (kB), 1024×1024 bytes in a megabyte (MB), and $1024 \times 1024 \times 1024$ bytes in a gigabyte (GB).

To store a bit we need a circuit which can `remember' a 0 or a 1. This is called a **bistable** circuit because it has two stable states. It can stay indefinitely either as a 0 or a 1. An example of a bistable circuit is shown in (Reference) (). It is made from two NOR gates.

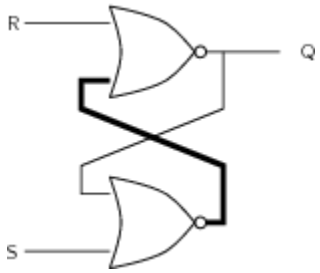

Figure 14

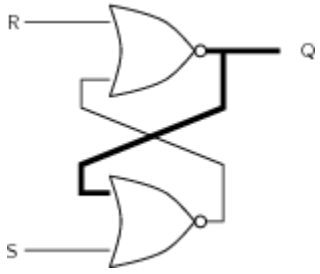

Figure 15

A bistable circuit made from two NOR gates. This circuit is able to store one bit of digital information. With the two inputs set to 0, you can see that the output could be (and will remain) either 0 or 1. The circuit on the top shows an output of 0, the circuit underneath shows an output of 1. Wires carrying high logic levels (1) are drawn thicker. The output of the bistable is labelled Q.

To store the 0 or the 1 in the bistable circuit, you set one of the inputs to 1, then put it back to 0 again. If the input labelled `S' (set) is raised, the output will immediately become 1. This is shown in **Figure 16** (#uid111).

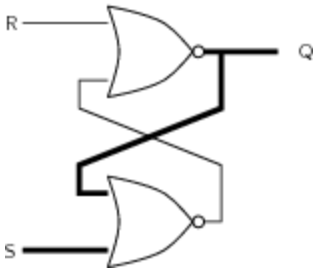

Figure 16: The output of a bistable circuit is **set** (made 1) by raising the `S' input to 1. Wires carrying high logic levels (1) are shown with thicker lines.

To store a 0, you raise the `R' (reset) input to 1. This is shown in **Figure 17** (#uid112).

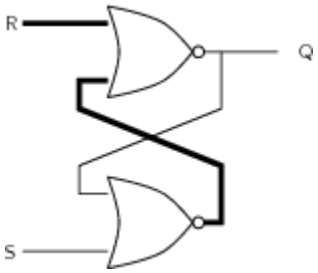

Figure 17: The output of a bistable circuit is **reset** (made 0) by raising the `R' input to 1. Wires carrying high logic levels (1) are shown with thicker lines.

Once you have used the S or R inputs to set or reset the bistable circuit, you then bring both inputs back to 0. The bistable `remembers' the state. Because of the ease with which the circuit can be Reset and Set it is also called an **RS flip flop** circuit.

Computer memory can store millions or billions of bits. If it used our circuit above, it would need millions or billions of NOR gates, each of which is made from several transistors. Computer memory is made of many millions of transistors.

NOTE: Interesting Fact :

The bistable circuits drawn here don't remember 0s or 1s forever — they lose the information if the power is turned off. The same is true for the RAM (Random Access Memory) used to store working and temporary data in a computer. Some modern circuits contain special memory which can remember its state even if the power is turned off. This is used in FLASH drives, commonly found in USB data sticks and on the memory cards used with digital cameras. These bistable circuits are much more complex.

You can also make T flip flops out of logic gates, however these are more complicated to design.

Counting Circuits

1. What is the term *bit* short for?
2. What is 43 in binary?
3. What is 1100101 in denary?
4. What is the highest number a modulo 64 counter can count to? How many T flip flops does it contain?
5. What is the difference between an RS flip flop and a T flip flop?
6. Draw a circuit diagram for a bistable circuit (RS flip flop). Make three extra copies of your diagram. On the first diagram, colour in the wires which will carry high voltage levels (digital 1) if the R input is low, and the S input is high. On the second diagram, colour in the wires which carry high voltage levels if the S input of the first circuit is now made low. On the third diagram, colour in the wires which carry high voltage levels if the R input is now made high. On the final diagram, colour in the wires carrying high voltage levels if the R input is now made low again.
7. Justify the statement: a modern computer contains millions of transistors.

End of Chapter Exercises

1. Calculate the reactance of a 3 mH inductor at a frequency of 50 Hz.
2. Calculate the reactance of a 30 μF capacitor at a frequency of 1 kHz.
3. Calculate the impedance of a series circuit containing a 5 mH inductor, a 400 μF capacitor and a 2 kΩ resistor at a frequency of 50 kHz.
4. Calculate the frequency at which the impedance of the circuit in the previous question will be the smallest.
5. Which component can be used to block low frequencies?
6. Draw a circuit diagram with a battery, diode and resistor in series. Make sure that the diode is forward biased so that a current will flow through it.
7. When building a complex electronic circuit which is going to be powered by a battery, it is always a good idea to put a diode in series with the battery. Explain how this will protect the circuit if the user puts the battery in the wrong way round.

8. Summarize the differences betwen a bipolar and field effect transistor.
9. What does an operational amplifier (op-amp) do?
10. What is the difference between a digital signal and an analogue signal?
11. What are the advantages of digital signals over analogue signals?
12. Draw the symbols for the five logic gates, and write down their truth tables.
13. Draw a circuit diagram with an AND gate. Each input should be connected to the output of a separate NOT gate. By writing truth tables show that this whole circuit behaves as a NOR gate.
14. Convert the denary number 99 into binary.
15. Convert the binary number 11100111 into denary.
16. Explain how three T flip flops can be connected together to make a modulo 8 counter. What is the highest number it can count up to?
17. Draw the circuit diagram for an RS flip flop (bistable) using two NOR gates.
18. Show how the circuit you have just drawn can have a stable output of 0 or 1 when both inputs are 0.
19. Operational (and other) amplifiers, logic gates, and flip flops all contain transistors, and would not work without them. Write a short newspaper article for an intelligent reader who knows nothing about electronics. Explain how important transistors are in modern society.